دانشگاه صنعتی شریف

دانشکده مهندسی برق

آزمایشگاه پیشرفته برنامه‌نویسی

"الگوریتم و شی‌گرایی"

نیم‌سال دوم (بهار) 00-01

آزمایش سوم

مقدمه

مسائل بسیاری در علوم کامپیوتر وجود دارد که برای حل آنها از الگوریتم های سرچ و انواع گراف‌ها استفاده می‌شود. در این آزمایشگاه می‌خواهیم در قالب پیاده سازی سلسله ای از کلاس ها برای کار با گراف، با مفاهیم شی گرایی در جاوا بیشتر آشنا شویم.

کلاس‌ها و جزئیات

هدف این آزمایشگاه، پیاده‌سازی الگوریتم DFS با استفاده از ویژگی شی‌گرایی جاوا می‌باشد. بدین منظور لازم است کلاس‌های زیر با مشخصات مشروح، تعریف گردد:

گراف به صورت مجموعه ای از یال‌ها و راس‌ها تعریف می‌شود. هر یال می‌تواند جهت‌دار یا بدون جهت باشد. می‌توانیم به هر یال ویژگی‌های مختلفی مثل یک عدد یا رنگ نسبت بدهیم. فعلا بدون در نظر گرفتن چنین ویژگی‌هایی کلاس‌های زیر را تعریف کنیم:

ابتدا دو کلاس DirectedEdge و UndirectedEdge تعریف کنیم که از کلاس کلی تری به نام Edge ارث می‌برند. حال اگر بخواهیم یال را با دو راسی که به آن متصل است مشخص کنیم لازم است کلاس Node را تعریف کنیم.

امتیازی: کلاس Edge را به صورت abstract تعریف کنید زیرا هر یال یا جهت دارد و یا بدون جهت است و دلیلی ندارد کسی بخواهد یالی بسازد بدون این که ویژگی جهت دار بودن یا نبودن را برایش ذکر کرده باشد.

کلاس Node شامل فیلدهای یک arraylist از Edge های متصل به این Node می‌باشد. متدهای این کلاس عبارتند از getInDegree و getOutDegree، متدی برای اضافه یا حذف کردن یال و متد getEdgesکه لیست همه یال‌های متصل به گره را بر می‌گرداند.

در حساب کردن درجه راس‌ها، یال‌های بدون جهت را به صورت دو یال جهت‌دار در جهت مخالف تعریف می‌کنیم. یعنی یک یال بدون جهت به Indegree و Outdegree یکی اضافه می کند.

پس از تعریف کلاس Node می توانیم فیلدهای کلاس DirectedEdge را مشخص کنیم. این کلاس شامل دو Node مبدا و مقصد است. همچنین UndirectedEdge شامل یک آرایه از Nodeها به طول 2 است که دو سر این یال را مشخص می‌کند.

هر کدام از این دو کلاس دو constructor دارند. یکی بدون ورودی و دیگری دو Node را به عنوان ورودی می گیرد. لازم نیست چک شود که این دو Node یکی نباشند (به عبارتی وجود طوقه مجاز است.) در نهایت نیز متدهای setter و getter را برای این دو کلاس می‌نویسیم.

حال پس از تعریف کلاس‌های مربوط به راس و یال، کلاس Graph را می‌توانیم به شکل زیر تعریف کنیم:

کلاس Graph شامل یک arraylist ازNodeها و یک arraylist از Edgeها است.

این کلاس متدهایی برای اضافه یا حذف کردن یال‌ها و راس‌ها دارد و همچنین متد getNodes و getEdgesکه همه ی راس‌ها و یال‌های گراف را بر می‌گرداند. اگر کاربر بخواهد از طریق متدی که برای اضافه کردن یال در نظر گرفته شده است، یالی را به گراف اضافه کند که راس هایش قبلا جزو Nodeهای گراف نبوده است، nodeهای جدید باید به مجموعه Nodeهای موجود اضافه شود.

توجه: اضافه کردن یک Node به تنهایی و به صورت isolated به گراف مجاز است و لزومی ندارد یالی به آن متصل باشد.

کلاس Graph دو constructor دارد که یکی بدون ورودی و دیگری لیستی از Edgeها به عنوان ورودی می‌گیرد.

حال زیرکلاس‌های گراف را تعریف میکنیم:

- زیرکلاس DirectedGraph از کلاس Graph ارث می‌برد با این تفاوت که arraylist یال‌ها تنها شامل DirectedEdge ها است.

حال باید دقت شود که این کلاس از کلاس Graph ارث می‌برد که به صورت پیش‌فرض متدی برای اضافه کردن Edge دارد. ممکن است کسی با استفاده از این متد تلاش کند یک UndirectedEdge به یال‌های این کلاس اضافه کند. برای جلوگیری از این اتفاق حتما متد addEdge کلاس پدر را باید Overrideکرد.

- زیرکلاس Tree از کلاس DirectedGraph ارث می برد با این تفاوت که Indegree هریک از nodeهای این کلاس حداکثر می تواند 1 باشد اما محدودیتی برای Outdegree وجود ندارد.

توجه شود که درخت، گرافی همبند است که وجود طوقه در آن مجاز نیست. متدهای DirectedGraph باید با توجه به این نکات Override شوند. همچنین در درخت هر راس حداکثر یک پدر دارد و بی‌شمار فرزند می‌تواند باشد. متدهای این کلاس شامل getFather و getChildren می‌باشد. حال متد getAncestors که پدر پدر راس و به همین ترتیب همه ی پدرهای موجود تا ریشه گراف را در یک لیست به ترتیب بر می‌گرداند را پیاده‌سازی کنید.

حال برای کلاس Tree یک متد به نام getPath بنویسید که دو Node به عنوان ورودی بگیرد و در صورتی که مسیری بین این دو راس در درخت وجود داشت، مسیر را به صورت لیستی از یال‌های این مسیر برگردانده و اگر مسیری یافت نشد، خروجی null برگرداند.

برای نوشتن این متد می توانید از متد getAncestors و یا از الگوریتم DFS استفاده کنید. توجه شود اگر برای حل این مساله بخواهید از الگوریتم DFS استفاده کنید احتمالا باید یک Boolean به نام visited به کلاس Node اضافه کنید.